Powered by
Futurense

BS./BSC.IN
Applied ai and Data Science

Linear Algebra and numerical analysis

Key Agenda items

Part I

- What is linear algebra, and why is it relevant and important to machine learning?
- How to create, index, and generally manipulate data in NumPy arrays.
- What a vector is and how to perform vector arithmetic and calculate vector norms.

Part II

- What a matrix is and how to perform matrix arithmetic, including matrix multiplication.
- A suite of types of matrices, their properties, and advanced operations involving matrices.
- Matrix factorization methods, including the eigen decomposition and singular-value decomposition.

Topics and learning objective

We will explore the following concepts in this video

- ❖ Introduction to linear Algebra and the concept of vectors
- ❖ System of linear equations

Learning objectives

- Students should be able to solve their assignments on linear systems and their applications in real-world applications after completing this video.

- Systems of linear equations and their solutions constitute one of the major topics we will study in this course. This first section will introduce some basic terminology and discuss a method for solving such systems.

More details about Linear Algebra

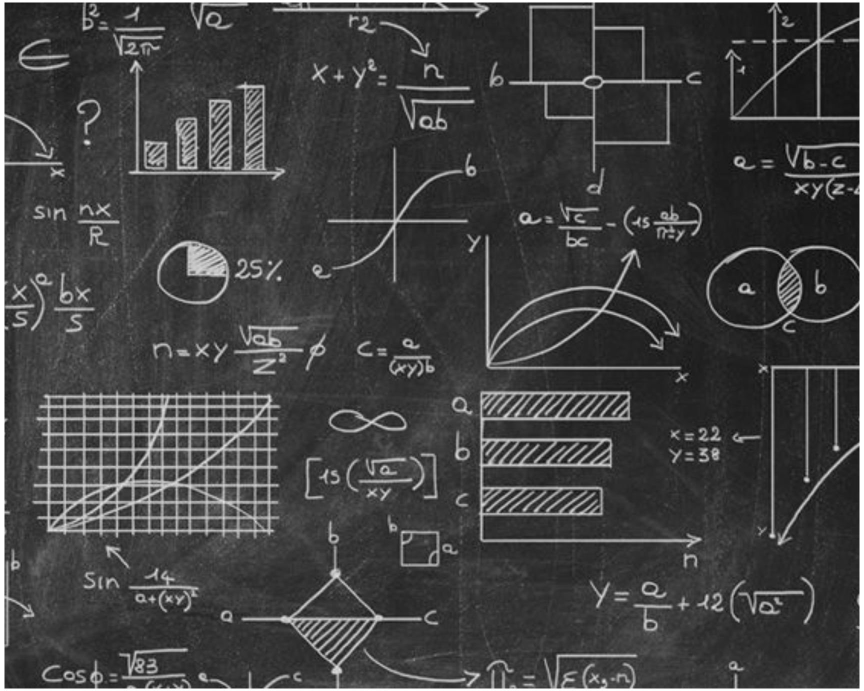
- Linear algebra is a branch of mathematics, but the truth of it is that linear algebra is the mathematics of data. Matrices and vectors are the language of data. Linear algebra is about linear combinations.
- Linear algebra is the study of lines and planes, vector spaces, and mappings that are required for linear transforms.
- It is a relatively young field of study, having initially been formalized in the 1800s to find unknowns in systems of linear equations.

History of Linear Algebra

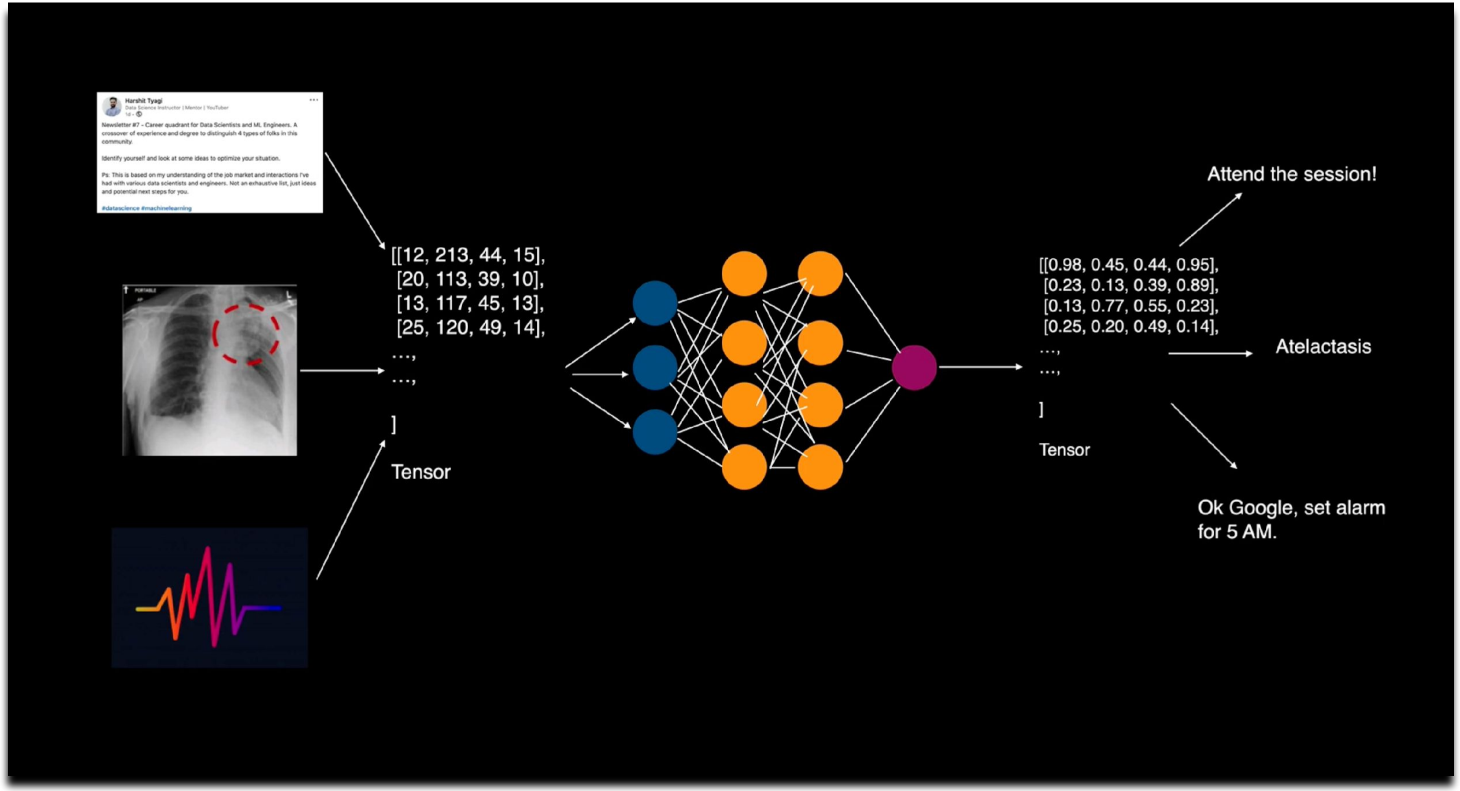
The study of linear algebra first emerged from the introduction of determinants. Determinants were considered by Leibniz in 1693, and subsequently, in 1750, Gabriel Cramer used them for giving solutions of linear systems, now called Cramer's Rule. Later, Gauss further developed the theory of solving linear systems by using Gaussian elimination. The study of matrix algebra first emerged in England in the mid-1800s. Linear algebra first appeared in American graduate textbooks in the 1940s and in undergraduate textbooks in the 1950s

Wide-Ranging applications of Linear Algebra in modern science

When you take a digital photo with your phone or transform the image in Photoshop, play a video game or watch a movie with digital effects, do a web search, or make a phone call, you use technologies that build upon linear algebra. Linear algebra is built on two basic elements, the matrix and the vector.



How Machine Learning uses Linear Algebra to solve data problems



Newsletter #7 - Career quadrant for Data Scientists and ML Engineers. A crossover of experience and degree to distinguish 4 types of folks in this community.

Identify yourself and look at some ideas to optimize your situation.

Ps: This is based on my understanding of the job market and interactions I've had with various data scientists and engineers. Not an exhaustive list, just ideas and potential next steps for you.

#datascience #machinelearning

Understanding concepts of vectors and Creating and Visualizing Vectors in Numeric Python

- In linear algebra, a vector is an ordered list of numbers.
- In abstract linear algebra, vectors may contain other mathematical objects including functions; however, as we are focused on applications, we will only consider vectors comprising numbers.

Vector Notation and formal definition

- a *vector* is an ordered list of numbers
- written as

$$\begin{bmatrix} -1.1 \\ 0.0 \\ 3.6 \\ -7.2 \end{bmatrix} \quad \text{or} \quad \begin{pmatrix} -1.1 \\ 0.0 \\ 3.6 \\ -7.2 \end{pmatrix}$$

  or $(-1.1, 0, 3.6, -7.2)$
- numbers in the list are the *elements* (*entries*, *coefficients*, *components*)
- number of elements is the *size* (*dimension*, *length*) of the vector
- vector above has dimension 4; its third entry is 3.6
- vector of size n is called an n-*vector*
- numbers are called *scalars*

Reference: Stanford ENGR108: Introduction to Applied Linear Algebra

Zero, ones and unit vectors

- n-vector with all entries 0 is denoted 0_n or just 0
- n-vector with all entries 1 is denoted $\mathbf{1}_n$ or just $\mathbf{1}$
- a *unit vector* has one entry 1 and all others 0
- denoted e_i where i is entry that is 1
- unit vectors of length 3:

$$e_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad e_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \quad e_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

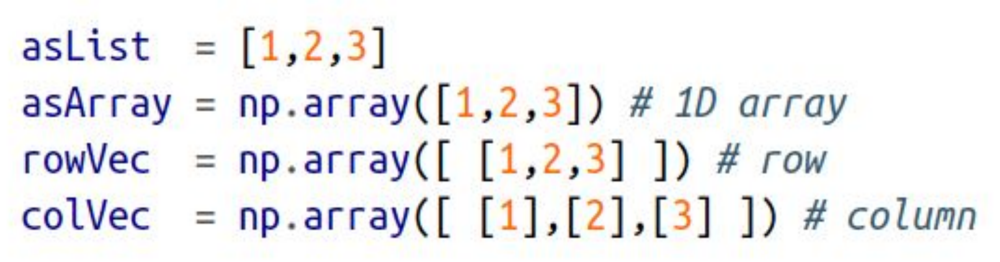
Reference: Stanford ENGR108: Introduction to Applied Linear Algebra

Important consideration for Python programming of vectors

- I introduced the dimensionality of a vector is the number of elements in that vector.
- However, in Python, the dimensionality of a vector or matrix is the number of geometric dimensions used to print out a numerical object.
- For example, all of the vectors shown above are considered “two-dimensional arrays” in Python, regardless of the number of elements contained in the vectors (which is the mathematical dimensionality).
- A list of numbers without a particular orientation is considered a 1D array in Python, regardless of the number of elements.
- The mathematical dimensionality—the number of elements in the vector—is called the length or the shape of the vector in Python.

Pythoning vectors using numpy arrays

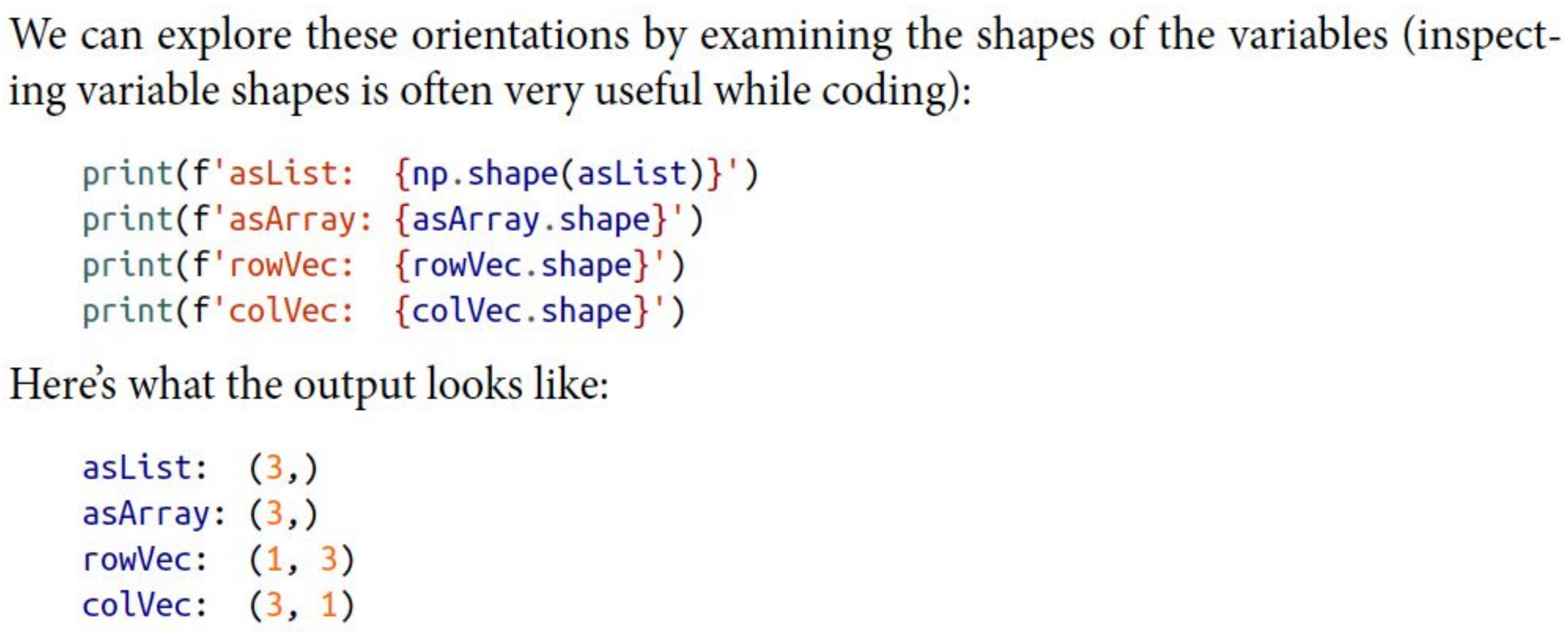
- *Vectors in Python can be represented using several data types.*
- *The list type may seem like the simplest way to represent a vector—and it is for for some applications.*
- *However, many linear algebra operations won't work on Python lists.*
- *Therefore, most of the time it's best to create vectors as NumPy arrays. The following code shows four ways of creating a vector:*



```
asList  = [1,2,3]
asArray = np.array([1,2,3]) # 1D array
rowVec  = np.array([ [1,2,3] ]) # row
colVec  = np.array([ [1],[2],[3] ]) # column
```

The variable `asArray` is an orientationless array, meaning it is neither a row nor a column vector but simply a 1D list of numbers in NumPy.

Exploring vector orientations by examining shape of the variables



We can explore these orientations by examining the shapes of the variables (inspecting variable shapes is often very useful while coding):

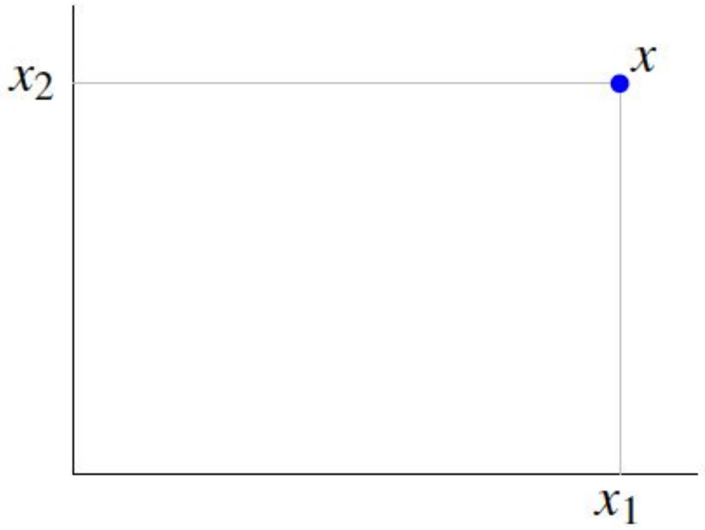
```
print(f'asList:  {np.shape(asList)}')
print(f'asArray: {asArray.shape}')
print(f'rowVec:  {rowVec.shape}')
print(f'colVec:  {colVec.shape}')
```

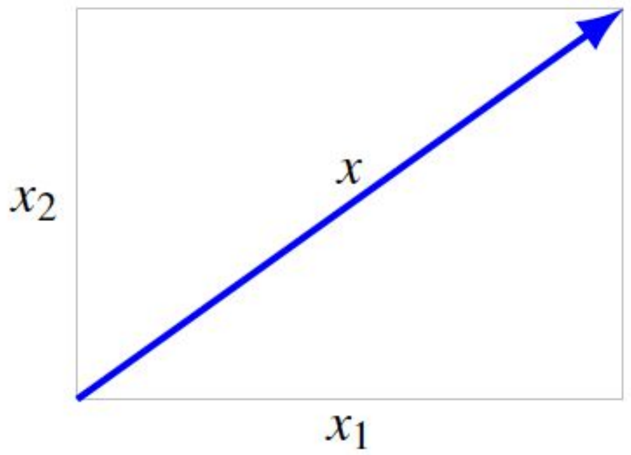
Here's what the output looks like:

```
asList:  (3,)
asArray: (3,)
rowVec:  (1, 3)
colVec:  (3, 1)
```

Location or displacement in 2D and 3D

2-vector (x1,x2) can represent a location or a displacement in 2-D

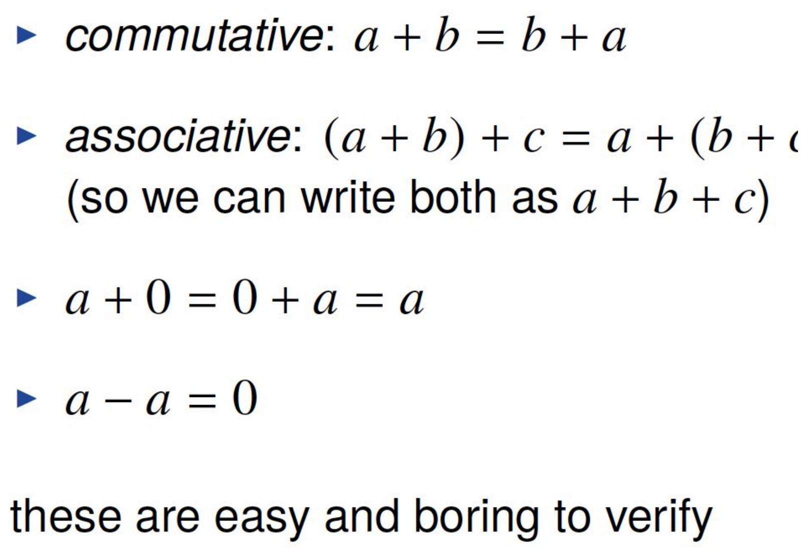




Numerous real-life applications of vector concept

- color: $(\mathrm{R}, \mathrm{G}, \mathrm{B})$
- quantities of n different commodities (or resources), *e.g.*, bill of materials
- portfolio: entries give shares (or $ value or fraction) held in each of n assets, with negative meaning short positions
- cash flow: x_i is payment in period i to us
- audio: x_i is the acoustic pressure at sample time i (sample times are spaced 1/44100 seconds apart)
- features: x_i is the value of ith *feature* or *attribute* of an entity
- customer purchase: x_i is the total $ purchase of product i by a customer over some period
- word count: x_i is the number of times word i appears in a document

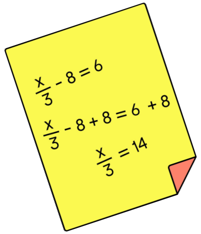
Operations on Vectors: Vector addition



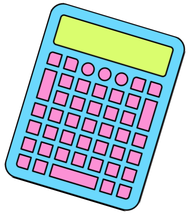
- *commutative*: $a + b = b + a$
- *associative*: $(a + b) + c = a + (b +$
  (so we can write both as $a + b + c$)
- $a + 0 = 0 + a = a$
- $a - a = 0$

these are easy and boring to verify

- Vectors are like nouns; they are the characters in our linear algebra story.
- The fun in linear algebra comes from the verbs—the actions that breathe life into the characters.
- Those actions are called operations.



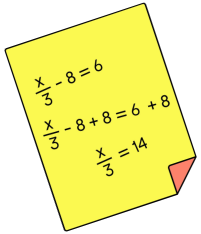
Quiz 1



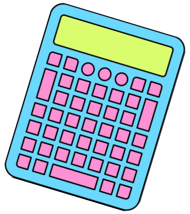
(a) **Given the following two vectors, what should be the result of the vector addition?**

(b) **Can you write down a Python code using Numpy to add s?**

$$\begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix} + \begin{bmatrix} 10 \\ 20 \\ 30 \end{bmatrix}$$



Quiz 2



Can you add a row vector to a column vector as in the following example?

$$\begin{bmatrix} 4 \\ 5 \\ 6 \end{bmatrix} + \begin{bmatrix} 10 & 20 & 30 \end{bmatrix} = ?$$

Scalar and vector multiplication

- scalar β and n-vector a can be multiplied

$$\beta a = (\beta a_1, \ldots, \beta a_n)$$

- also denoted $a\beta$
- example:

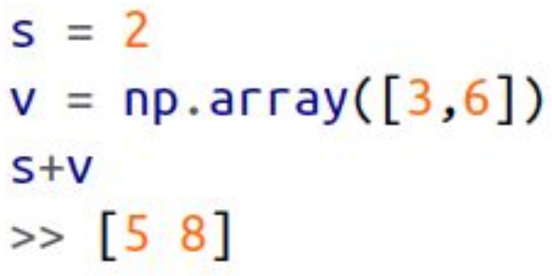
$$(-2)\begin{bmatrix} 1 \\ 9 \\ 6 \end{bmatrix} = \begin{bmatrix} -2 \\ -18 \\ -12 \end{bmatrix}$$

Scalar and vector addition

- Adding a scalar to a vector is not formally defined in linear algebra: they are two separate kinds of mathematical objects and cannot be combined.

- However, numerical processing programs like Python will allow adding scalars to vectors, and the operation is comparable to scalar-vector multiplication: the scalar is added to each vector element.

The following code illustrates the idea:

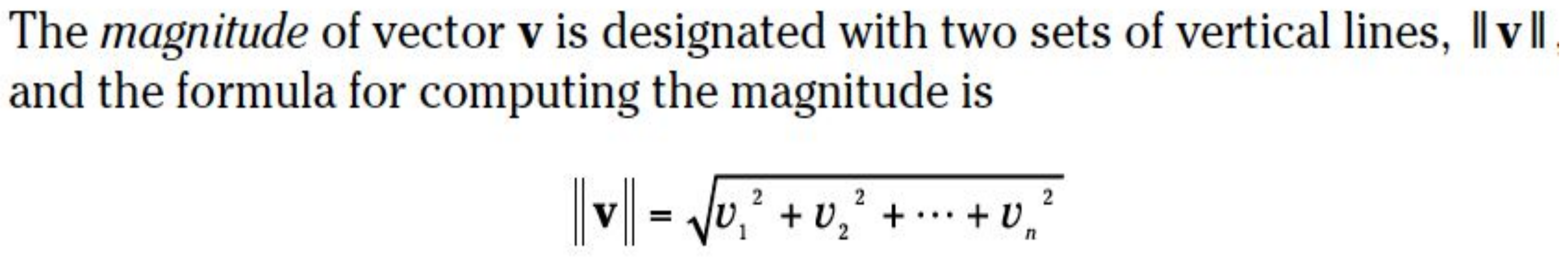


```
s = 2
v = np.array([3,6])
s+v
>> [5 8]
```

Estimation of Vector Magnitude

- The magnitude of a vector is also referred to as its length or norm.

- In two-space or three-space, you see a ray that has a particular length and can visualize where the magnitude's numerical value



The *magnitude* of vector **v** is designated with two sets of vertical lines, $\|\mathbf{v}\|$, and the formula for computing the magnitude is

$$\|\mathbf{v}\| = \sqrt{v_1^2 + v_2^2 + \cdots + v_n^2}$$

Example of Estimation of Vector Magnitude

For example, with the following vector

$$\mathbf{v} = \begin{bmatrix} 3 \\ 2 \\ 4 \end{bmatrix}$$

the magnitude is found with:

$$\|\mathbf{v}\| = \sqrt{3^2 + 2^2 + 4^2} = \sqrt{9 + 4 + 16} = \sqrt{29} \approx 5.4$$

$$\mathbf{v} = \begin{bmatrix} 1 \\ -4 \\ -2 \\ 2 \end{bmatrix}$$

When you compute the magnitude, you get

$$\|\mathbf{v}\| = \sqrt{1^2 + (-4)^2 + (-2)^2 + 2^2} = \sqrt{1 + 16 + 4 + 4} = \sqrt{25} = 5$$

Example of Estimation of Vector Magnitude by a scalar multiplication

Now multiply the same vector v in the previous example by k = 3, 3v, and compute the magnitude of the resulting vector.

$$3\mathbf{v} = 3\begin{bmatrix} 1 \\ -4 \\ -2 \\ 2 \end{bmatrix} = \begin{bmatrix} 3 \\ -12 \\ -6 \\ 6 \end{bmatrix} \text{ and } \|3\mathbf{v}\| = \sqrt{3^2 + (-12)^2 + (-6)^2 + 6^2}$$

$$= \sqrt{9 + 144 + 36 + 36} = \sqrt{225} = 15$$

Adjusting magnitude for scalar multiplication

The magnitude of the product of a scalar, k, and a vector, v, is equal to the absolute value of the scalar and the magnitude of the original vector, |k|·||v||.

And now, to show you that I'm not fooling, here's how the math works:

$$k\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ \vdots \\ v_n \end{bmatrix} = \begin{bmatrix} kv_1 \\ kv_2 \\ kv_3 \\ \vdots \\ kv_n \end{bmatrix} \text{ so } \|k\mathbf{v}\| = \sqrt{(kv_1)^2 + (kv_2)^2 + \cdots + (kv_n)^2}$$

$$= \sqrt{k^2v_1^2 + k^2v_2^2 + \cdots + k^2v_n^2}$$

$$= \sqrt{k^2\left(v_1^2 + v_2^2 + \cdots + v_n^2\right)}$$

$$= \sqrt{k^2}\sqrt{\left(v_1^2 + v_2^2 + \cdots + v_n^2\right)}$$

$$= |k|\sqrt{v_1^2 + v_2^2 + \cdots + v_n^2}$$

$$= |k| \cdot \|\mathbf{v}\|$$

Adjusting magnitude for scalar multiplication

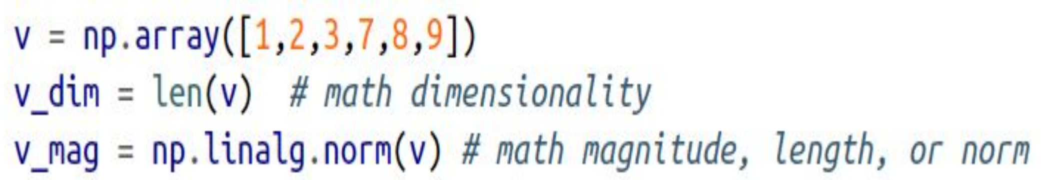
The magnitude of kv is equal to the absolute value of k times the magnitude of the original vector, v. The square root of a square is equal to the absolute value of the number that’s being squared:

$$\sqrt{k^2} = |k|$$

Using the absolute value takes care of the instances when k is a negative number. So, if you multiply a vector by a negative number, the value of the magnitude of the resulting vector is still going to be a positive number.

Pythoning Vector Magnitude

- **Before showing the Python code, let me explain some more terminological discrepancies between "chalkboard" linear algebra and Python linear algebra.**
- **In mathematics, the dimensionality of a vector is the number of elements in that vector, while the length is a geometric distance; in Python, the function len() (where len is short for length) returns the dimensionality of an array, while the function np.norm() returns the geometric length (magnitude).**

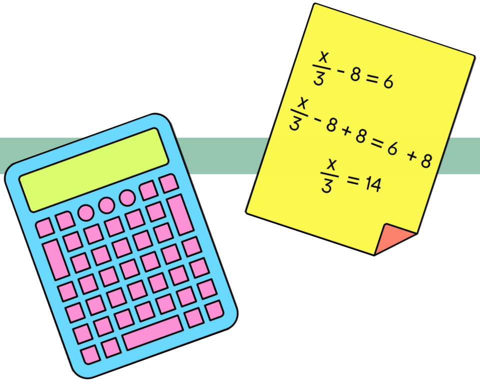


```
v = np.array([1,2,3,7,8,9])
v_dim = len(v)  # math dimensionality
v_mag = np.linalg.norm(v) # math magnitude, length, or norm
```

There are some applications where we want a vector that has a geometric length of one, which is called a unit vector. Example applications include orthogonal matrices, rotation matrices, eigenvectors, and singular vectors.

A unit vector is defined as $\| \mathbf{v} \| = 1$.

Wrap-up



- We learned that linear algebra methods are indispensable to modern data science and AI
- In this course, we will only consider vectors comprising numbers and they are key to understanding machine learning problems.
- We have learned Python syntax and the use of Numpy to create, index, and generally manipulate data in NumPy arrays.
- Finally, we comprehensively learned about vector operations and how to perform vector arithmetic and calculate vector norms.

.

THANK YOU!

In the next video:

Introduction to Linear Equations and Solvers